Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 164 (2006) 177–194
www.elsevier.com/locate/entcs

A Coalgebraic Representation of Reduction by Cone of Influence

Hiroshi Watanabe[1,2] Koki Nishizawa[3] Osamu Takaki[1,4,5]

Research Center for Verification and Semantics
National Institute of Advanced Industrial Science and Technology (AIST)
Mitsui Sumitomo Kaijo Senri Bldg., 1-2-14 Shin-Senri Nishi
Toyonaka, Osaka 560-0083, Japan

Abstract

The Cone of Influence Reduction is a fundamental abstraction technique for reducing the size of models used in symbolic model checking. We develop coalgebraic representations of systems as composites of state transition maps and connectors. These representations include synchronous systems, asynchronous systems, asynchronous systems with synchronization by channels, and those with shared variables, probabilistic synchronous systems and so on. We schematically show the cone of influence reduction using these coalgebraic representations, which give a unified framework for providing the technique for various kinds of systems.

Keywords: coalgebra, cone of influence reduction, abstraction, model checking

1 Introduction

The Cone of Influence Reduction (COI for short) [3] is a fundamental abstraction technique for reducing the size of models used in symbolic model checking. It is also called slicing in [5] or localization reduction in [6]. The idea of COI is simple. Suppose a model (Kripke structure) is specified by some state variables. The COI reduces the size of the model by getting rid of redundant variables which are not related to the property we want to verify. These redundant variables are chosen by analysing the dependency graph of state space variables specifying the model. By COI, we obtain a bisimulation or simulation between the original model and the reduced model. Since bisimulation and simulation are relations that preserve

[1] The authors acknowledge the support from MEXT Japan through Special Coordination Funds for Promoting Science and Technology.

[2] Email: hirowata@ni.aist.go.jp

[3] Email: koki-nishizawa@aist.go.jp

[4] Email: o-takaki@aist.go.jp

[5] Moved from Kyoto Sangyo University

doi:10.1016/j.entcs.2006.06.011

the formulas of CTL^* and $ACTL^*$ (or $ECTL^*$) [2], respectively, COI works as an abstraction technique.

Our motivations of this paper come from the textbook by Clarke et al.[3], where the authors introduced the COI technique for synchronous transition systems (synchronous circuits) by showing, in the framework of Boolean expressions of transition systems, that a projective relation gives a bisimulation between the original and the reduced systems. The proof recalls coalgebras. Our first motivation is to understand the abstraction technique of the COI in a coalgebraic framework.

Another crude motivation is given by the question: Can COI be applied to various formalisms of specifications used in various model checkers? There are various specification languages used in model-checkers to describe the behaviour of models, e.g., synchronous systems and asynchronous systems in SMV [13], networks of timed automata communicating through channels and shared data structures in UPPAAL [15], and probabilistic transition systems in PRISM [10] etc. The idea of COI is so simple and natural that it seems applicable to all of these systems, but COI results in yielding bisimulation or simulation are not clear in individual cases. Coalgebras may help for sorting out this question.

In this paper, we develop coalgebraic representations of systems. When we specify models of large complex systems, we use, in many cases, large numbers of state variables. It is sometimes possible to specify the macroscopic behaviour of such models by combining the behaviours of each state variable specified by a state transition map for individual variables We are interested in the cases where models can be characterised by coalgebras with carriers given by cartesian products of domains of individual state variables, and structure maps given by composites of state transition maps and connectors. These models include synchronous systems, asynchronous systems, and so forth.

Our main theorem is presented in Section 5.2, where we give the COI reduction from the viewpoint of coalgebraic representation. We represent the COI reduction separately as the existence of a reduced coalgebra and the property that the projection gives a morphism/lax morphism/oplax morphism from the original coalgebra to the reduced coalgebra. The advantage of the coalgebraic approach is that the main theorem, the COI, is schematically shown by two diagrams, one arising from analysis of the dependency graph and the other from analysis of the connector. The possibility of COI and its result for bisimulation or simulation depends on the analysis of the connectors. This point of view gives a simple, unified framework for providing COI reduction for various formalisms. We can obtain COI results for synchronous, asynchronous, asynchronous with communication, and those formalisms with shared data structures and probabilistic synchronous systems, with and without guards on transitions.

Moreover, we consider higher level constructs that compose systems and COI results. We illustrate the coalgebraic approach for presenting compositions, exhibiting two nondeterministic choice compositions that preserve COI reductions, by following a nondeterministic choice composition of modules in SMV.

We believe this approach may help non-expert users of model checkers to think

about and improve their unchecked specifications, as well as designers of specification language for tools to provide the pre-process for model check in various frameworks.

The textbook by Clarke et al. [3], introduces the COI and a bisimulation result between the original transition system and the reduced one, in the case of synchronous systems where all state variables update synchronously. The paper [1] reports a COI reduction for Bounded Model Checking. Various case studies (for example [17]) have used COI. The tool NuSMV2 [9] implemented the COI as an optional pre-process for model checking. The idea of connectors comes from a definition of a monoidal functor and a monoidal composition in category theory.

We proceed as follows. In Section 2, we recall the notion of coalgebras and their morphisms. We represent the transition systems and probabilistic transition systems as coalgebras. In Section 3, we introduce the notion of connectors which combine the behaviours of all variables. We introduce a property of connectors. In Section 4, we represent various systems through a composite of state transition maps and a connector. In Section 5, we show the cone of influence reduction from the view point of coalgebraic representation. In Section 6, we show the analysis of a dependency graphs. We exhibit two constructions which preserve COI reduction in section 7.

2 Transition Systems and Coalgebras

We recall the definition of transition systems and coalgebras, and the relationship between them.

2.1 Transition Systems

Definition 2.1 A *transition system* is a pair (S, T) where S is a set of states, and $T \subseteq S \times S$ is a binary relation on S, called *transition relation*. When $(s, s') \in T$, we write $s \to s'$, and interpret a transition from state s to s'.

A transition system (S, T) is called *non-terminating* (or *deadlock-free*) if the transition relation T is total, i.e., $\{s' | (s, s') \in T\} \neq \emptyset$ for every $s \in S$.

2.2 Simulations and Bisimulations

Definition 2.2 Let $M_1 = (S_1, T_1)$, $M_2 = (S_2, T_2)$ be transition systems.

- A binary relation $R \subseteq S_1 \times S_2$ is called *a simulation between M_1 and M_2 (or M_2 simulates M_1 by R)* if it satisfies the following condition: When $(s_1, s_2) \in R$,
  - for any transition $s_1 \to s_1'$ in M_1, there is a transition $s_2 \to s_2'$ in M_2 such that $(s_1', s_2') \in R$.
- A binary relation $R \subseteq S_1 \times S_2$ is called a *bisimulation between M_1 and M_2* if it satisfies the following two conditions:

(i) R is a simulation between M_1 and M_2,

(ii) the converse relation $R^{-1} (= \{(s_2, s_1) | (s_1, s_2) \in R\})$ is also a simulation between M_2 and M_1.

2.3 Coalgebras

For the introductory reference of coalgebras, please refer to Rutten's paper [11].

Let Set be the category of sets and functions, $F : \mathsf{Set} \to \mathsf{Set}$ an endofunctor on Set.

Definition 2.3 An F-coalgebra is a pair $(S, \alpha : S \to FS)$ consisting of a set S, called a *carrier*, and a map $\alpha : S \to FS$, called a *structure map*.

We give two main examples; $\mathcal{P}$-coalgebras for powerset functor $\mathcal{P}$, and $\mathcal{D}$-coalgebras for probability distribution functor $\mathcal{D}$.

$\mathcal{P}$-coalgebras as transition systems

Let $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ be the *powerset functor* which sends each set S to its powerset $\mathcal{P}S = \{X | X \subseteq S\}$, and each map $f : S \to S'$ to a map $\mathcal{P}f : \mathcal{P}S \to \mathcal{P}S'$ defined by $\mathcal{P}f(X) = \{f(x) | x \in X\}$ for $X \in \mathcal{P}S$. A $\mathcal{P}$-coalgebra is a pair $(S, \alpha : s \to \mathcal{P}S)$ consisting of a carrier S with a structure map $\alpha : S \to \mathcal{P}S$. Since there is a bijection between the set of all maps from S to $\mathcal{P}S$ and the set of all binary relations on S, every $\mathcal{P}$-coalgebra (S, α) is equivalent to a transition system (S, T) under the correspondence:

$$(s, s') \in T \Leftrightarrow s' \in \alpha(s). \tag{1}$$

Hence, we identify a $\mathcal{P}$-coalgebra with a transition system in this paper.

For the case of *nonempty powerset functor* $\mathcal{P}_+ : \mathsf{Set} \to \mathsf{Set}$ which sends set S to its powerset without an empty set, a $\mathcal{P}_+$-coalgebra is equivalent to a non-terminating transition system.

$\mathcal{D}$-coalgebras as probabilistic transition systems

A *probability distribution over* a set X is a function $\mu : X \to [0, 1]$ with support finite (or at most countable) such that $\Sigma_{x \in X} \mu(x) = 1$ holds. We say a function $\mu : X \to [0, 1]$ is *support finite* if the set $\{x | f(x) \neq 0\}$ of non-zero elements is finite. For given two probability distributions μ, ν over sets X, Y, respectively, the *product probability distribution* $\mu \times \nu$ over the $X \times Y$ is defined by $\mu \times \nu(x, y) = \mu(x) \cdot \nu(y)$ for $\langle x, y \rangle \in X \times Y$.

Let $\mathcal{D} : \mathsf{Set} \to \mathsf{Set}$ be the probability distribution functor which sends a set X to a set $\mathcal{D}X$ of all probability distributions over the X, and a function $f : X \to Y$ to a function $\mathcal{D}f : \mathcal{D}X \to \mathcal{D}Y$ defined by $\mathcal{D}f(\mu)(y) = \sum_{x \in f^{-1}(y)} \mu(x)$ for $\mu \in \mathcal{D}X$, $y \in Y$. For a $\mathcal{D}$-coalgebra $(S, \alpha : S \to \mathcal{D}S)$, the structure map α assigns each state $x \in S$ to a probability distribution $\alpha(x)$ of next states. So, $\mathcal{D}$-coalgebras for the probability distribution functor $\mathcal{D}$ are called Markov chains, probabilistic transition systems or probabilistic automata, etc.

2.4 Morphisms of Coalgebras

Next, morphisms of coalgebras and their relationship to bisimulation and simulation are recalled:

Definition 2.4 Let (S_1, α_1), (S_2, α_2) be F-coalgebras.

- A map $f : S_1 \to S_2$ is called a *morphism of F-coalgebras* $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ when $F(f) \circ \alpha_1 = \alpha_2 \circ f$ holds.

Assume for the functor $F : \mathsf{Set} \to \mathsf{Set}$ that every image FX is equipped with a partial order $\leq_{FX}$.

- A map $f : S_1 \to S_2$ is called a *lax-morphism of F-coalgebras* $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ if $Ff \circ \alpha_1(s) \leq_{FS_2} \alpha_2 \circ f(s)$ holds for every $s \in S_2$.
- A map $f : S_1 \to S_2$ is called an *oplax morphism of F-coalgebras* $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ if $Ff \circ \alpha_1(s) \geq_{FS_2} \alpha_2 \circ f(s)$ holds for every $s \in S_2$.

We remark that

Remark 2.5 If a map $f : S_1 \to S_2$ is both lax and oplax morphism between (S_1, α_1) and (S_2, α_2), then the map f is a morphism between F-coalgebras.

2.4.1 Morphisms of $\mathcal{P}$-coalgebras

Morphisms of $\mathcal{P}$-coalgebras are related to bisimulation and simulation between transition systems.

Proposition 2.6 *Let (S_1, α_1) and (S_2, α_2) be $\mathcal{P}$-coalgebras. For every set X, fix a partial order on $\mathcal{P}(X)$ to be an usual inclusion relation $\subseteq$.*

- *If $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ is a morphism of $\mathcal{P}$-coalgebras, then its graph $\mathsf{Graph}(f) = \{(s_1, f(s_1)) | s_1 \in S_1\}$ gives a bisimulation between transition systems (corresponding to) (S_1, α_1) and (S_2, α_2).*
- *If $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ is a lax morphism, then its graph $\mathsf{Graph}(f)$ gives a simulation between (S_1, α_1) and (S_2, α_2).*
- *If $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ is an oplax morphism, then the converse relation $\mathsf{Graph}(f)^{-1}$ of the graph gives a simulation between (S_2, α_2) and (S_1, α_1).*

Let V be a set, and $\mathcal{P}_V : \mathsf{Set} \to \mathsf{Set}$ an endofunctor defined by $\mathcal{P}_V(X) = \mathcal{P}(X \times V)^V$. Every $\mathcal{P}_V$-coalgebra $(S, \alpha : S \to \mathcal{P}_V(S))$ is equivalent to a $\mathcal{P}$-coalgebra $(S \times V, \bar{\alpha})$ where $\bar{\alpha}$ is the exponential adjunct of α given by the bijective correspondence:

$$\frac{\alpha : S \longrightarrow \mathcal{P}(S \times V)^V}{\bar{\alpha} : S \times V \longrightarrow \mathcal{P}(S \times V).}$$

We use this lemma later.

Lemma 2.7 *Let Under the assumption that every image of $\mathcal{P}_V$ is equipped with a pointwise partial order given by inclusion.*

- *If $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ is a morphism of $\mathcal{P}_V$-coalgebras, then the map $f \times \mathrm{id}_V : (S_1 \times V, \bar{\alpha_1}) \to (S_2 \times V, \bar{\alpha_2})$ is a morphism of $\mathcal{P}$-coalgebras.*
- *If $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ is a lax/oplax morphism of $\mathcal{P}_V$-coalgebras, then the map $f \times \mathrm{id}_V : (S_1 \times V, \bar{\alpha_1}) \to (S_2 \times V, \bar{\alpha_2})$ is a lax/oplax morphism of $\mathcal{P}$-coalgebras.*

2.4.2 Morphisms of $\mathcal{D}$-coalgebras

If $f : (S_1, \alpha_1) \to (S_2, \alpha_2)$ is a morphism of $\mathcal{D}$-coalgebras, then the relation $\mathsf{Graph}(f)$ has the following property:

- For every $(s_1, f(s_1)) \in \mathsf{Graph}(f)$, $\alpha_1(s_1)(U) = \alpha_2(f(s_1))(V)$ holds for every pair $U \subseteq S_1$, $V \subseteq S_2$ such that $U = f^{-1}(V)$.

This relation $\mathsf{Graph}(f)$ is a bisimulation of $\mathcal{D}$-coalgebras in the sense of [11]. It is reported in [16] that a bisimulation of $\mathcal{D}$-coalgebras give a probabilistic bisimulation of Discrete Time Markov Chains in the sense of [8]. Hence the above $\mathsf{Graph}(f)$ gives a probabilistic bisimulation of Discrete Time Markov Chains.

2.5 State Transition Maps for Individual Variables

In order to specify a state of a system, we choose some variables that describe the properties of the system, and represent a state by a tuple of values for these variables. Next, we specify the state transitions of these variables. Here, we are interested in the state transitions specified by the state transition maps of individual variables.

Suppose a finite set I represents the set of variables, and a set S_i represents the state space for each variable $i \in I$. We obtain a collection $\{S_i\}_{i \in I}$ of state spaces of individual variables .

Definition 2.8 Let I be a finite set of state variables, $\{S_i\}_{i \in I}$ a finite family of sets representing state spaces for the state variables. A *state transition map of a state variable $j (\in I)$ for a functor $F : \mathsf{Set} \to \mathsf{Set}$* is a map:

$$f : \prod_{i \in I} S_i \to F(S_j).$$

A collection of state transition maps is called a *family of state transition maps.*

We give some examples of state transition maps:

Suppose we have three state variables x, y, and z which vary over the set S_{x},S_{y}, and S_{z}, respectively. For example, we can use state transition map for the identity functor:

$$S_{\mathsf{x}} \times S_{\mathsf{y}} \times S_{\mathsf{z}} \longrightarrow S_{\mathsf{x}} \tag{2}$$

to specify a deterministic transition of variable x that is determined by the current states of x, y, and z.

For the case of nondeterministic transitions, we use state transition maps for the powerset functor:

$$S_{\mathsf{x}} \times S_{\mathsf{y}} \times S_{\mathsf{z}} \longrightarrow \mathcal{P}(S_{\mathsf{x}}). \tag{3}$$

For the case of a nondeterministic transitions with at least one successor, we use state transition map for the nonempty powerset functor:

$$S_{\mathsf{x}} \times S_{\mathsf{y}} \times S_{\mathsf{z}} \longrightarrow \mathcal{P}_+(S_{\mathsf{x}}). \tag{4}$$

Example 2.9 Consider the following example of SMV code:

```
init(x) := 0;
next(x) := Case
    x = 0 & y = 0: 1;
    x = 1 & z = 0: {0,1};
    1 : 0;
```

This code defines the initial value of `x` to be 0, and next state of variable `x`. If the current state of `x` is 0 and y is also 0, then the next state of `x` is 1. If `x` is 1 and `z` is 0, then the next state of `x` is 0 or 1. Otherwise, the next state of `x` is 0. This rule can be represented in the transition map of the form (4).

For the case of probabilistic transitions, we use state transition map for the probability distribution functor:

$$S_{\mathsf{x}} \times S_{\mathsf{y}} \times S_{\mathsf{z}} \longrightarrow \mathcal{D}(S_{\mathsf{x}}). \tag{5}$$

We can model more complex behaviours. Consider the case of describing a network of automata which communicate with each other through channels and shared data structures, like some fragment of the modeling language of UPPAAL [15]. On each transition of the individual automaton, we may want to define

- a guard, which is a condition on the states of another automata and shared variables, enabling the transition,
- a synchronisation label, like input channel $a?$ or output channel $a!$, for synchronising transitions with another automata,
- a command, such as an assignment of values, which changes the state of shared variables.

In this case, assuming V stands for a domain of shared variables and C for a set of channels, we employ an endofuntor functor $B_{C,V} : \mathsf{Set} \to \mathsf{Set}$ defined by

$$B_{C,V}(X) = \mathcal{P}(X \times V + C \times X \times V + C \times X \times V)^V,$$

for a set X, and specify by the state transition map for the functor $B_{C,V}$:

$$f : S_{\mathsf{x}} \times S_{\mathsf{y}} \times S_{\mathsf{z}} \longrightarrow B_{C,V}(S_{\mathsf{x}}). \tag{6}$$

Here, we denote by $\kappa_i (i = 1, 2, 3)$ the i-th injection to the coproduct $S_{\mathsf{x}} \times V + C \times S_{\mathsf{x}} \times V + C \times S_{\mathsf{x}} \times V$. We describe

- $\kappa_1 \langle s'_{\mathsf{x}}, v' \rangle \in f(s_{\mathsf{x}}, s_{\mathsf{y}}, s_{\mathsf{z}})(v)$ iff there is a transition without label from s_{x} to s'_{x} with a guard which is satisfied by $s_{\mathsf{y}} \in S_{\mathsf{y}}$, $s_{\mathsf{z}} \in S_{\mathsf{z}}$, and $v \in V$, and the accompanying command updates shared data from v to v',
- $\kappa_2 \langle c, s'_{\mathsf{x}}, v' \rangle \in f(s_{\mathsf{x}}, s_{\mathsf{y}}, s_{\mathsf{z}})(v)$ iff there is a transition from s_{x} to s'_{x} with output channel $c!$ and a guard which is satisfied by $s_{\mathsf{y}} \in S_{\mathsf{y}}$, $s_{\mathsf{z}} \in S_{\mathsf{z}}$, and $v \in V$, and the command updates the shared data from v to v',
- $\kappa_3 \langle c, s'_{\mathsf{x}}, v' \rangle \in f(s_{\mathsf{x}}, s_{\mathsf{y}}, s_{\mathsf{z}})(v)$ iff there is a transition with input channel $c?$ from s_{x} to s'_{x} with a guard which is satisfied by $s_{\mathsf{y}} \in S_{\mathsf{y}}$, $s_{\mathsf{z}} \in S_{\mathsf{z}}$, and $v \in V$, and the

command updates the shared data from v to v'.

As we see above, we can specify the various kinds of transitions by state transition maps for appropriate functor F. The next question is how can we combine the transitions to model entire systems. We give some connectors that combine these state transition maps for individual variables.

3 Connectors

We give an abstract definition of connector and three concrete instances that construct systems, in order to demonstrate COI in Section 5.2. We also give a key property of these connectors.

Definition 3.1 Given endofunctors $G, H : \mathsf{Set} \to \mathsf{Set}$, a *connector* γ *from* G *to* H is a function which assigns each finite family $\{S_i\}_{i\in I}$ of sets to a map

$$\gamma_{\{S_i\}_{i\in I}} : \prod_{i\in I} G(S_i) \longrightarrow H(\prod_{i\in I} S_i) \tag{7}$$

which is natural for every $S_i (i \in I)$.

We give three concrete examples of connectors.

3.1 Synchronous Connector

Definition 3.2 A *synchronous connector* sync is a connector from $\mathcal{P}_+$ to $\mathcal{P}$ with components

$$\mathsf{sync}_{\{S_i\}_{i\in I}} : \prod_{i\in I} \mathcal{P}_+(S_i) \longrightarrow \mathcal{P}(\prod_{i\in I} S_i) \tag{8}$$

defined by

$$\langle A_i\rangle_{i\in I} \mapsto \prod_{i\in I} A_i.$$

Remark 3.3 In the case of singleton family $\{S\}$, the component $\mathsf{sync}_{\{S\}}$ is just the inclusion $\mathcal{P}_+(S) \to \mathcal{P}(S)$.

3.2 Asynchronous Connector

Definition 3.4 Let C be a set of channels and V a domain of shared variables. A (C, V)-*asynchronous connector* $\mathsf{async}^{C,V}$ is a connector from $\mathsf{Id} \times B_{C,V}$ to $\mathcal{P}_V$ with components

$$\mathsf{async}^{C,V}_{\{S_i\}_{i\in I}} : \prod_{i\in I} (S_i \times B_{C,V}(S_i)) \longrightarrow \mathcal{P}_V(\prod_{i\in I} S_i) \tag{9}$$

sending $\langle\langle s_i, \alpha_i\rangle\rangle_{i\in I}$ to a map

$$\lambda v \in V. \left\{ \langle\langle s'_i\rangle_{i\in I}, v'\rangle \;\middle|\; \begin{array}{c} \exists j \in I.\ \kappa_{j,1}\langle s'_j, v'\rangle \in \alpha_j(v), \\ s'_i = s_i (i \neq j) \end{array} \right\} \cup \left\{ \langle\langle s'_i\rangle_{i\in I}, v''\rangle \;\middle|\; \begin{array}{l} \exists c \in C, \exists j, l \in I.\ j \neq l, \\ \qquad \kappa_{j,2}\langle c, s'_j, v'\rangle \in \alpha_j(v), \\ \qquad \kappa_{l,3}\langle c, s'_l, v''\rangle \in \alpha_l(v'), \\ \qquad s'_i = s_i (i \neq j, l) \end{array} \right\}, \tag{10}$$

where $\kappa_{i,l} (i \in I, 1 \leq l \leq 3)$ is the l-th injection to the coproduct

$$S_i \times V + C \times S_i \times V + C \times S_i \times V.$$

The first set of the union in (10) describes the successors given by single transitions of automata, and the second set describes the successors given by communications between two different automata.

Remark 3.5 In the case of the singleton family, the component $\mathsf{async}^{C,V}_{\{S\}} : S \times B_{C,V}(S) \to \mathcal{P}_V(S)$ is defined by

$$\langle s, \alpha\rangle \mapsto \lambda v.\{\langle s', v'\rangle \mid \kappa_1\langle s', v'\rangle \in \alpha(v)\},$$

where κ_1 is the first injection to the coproduct $S \times V + C \times S \times V + C \times S \times V$.

Example 3.6 • In the case of $C = \emptyset$ and $V = 1$ the singleton, the component $\mathsf{async}^{\emptyset,1}_{\{S_1,S_2\}}$ is equivalent to a map from $(S_1 \times \mathcal{P}(S_1)) \times (S_2 \times \mathcal{P}(S_2))$ to $\mathcal{P}(S_1 \times S_2)$ given by

$$\langle\langle s_1, A_1\rangle, \langle s_2, A_2\rangle\rangle \mapsto \{\langle s_1, s'_2\rangle \mid s'_2 \in A_2\} \cup \{\langle s'_1, s_2\rangle \mid s_1 \in A_1\}.$$

• When $V = 1$, the component $\mathsf{async}^{C,1}_{\{S_1,S_2\}}$ is equivalent to a map

$$(S_1 \times \mathcal{P}(S_1 + C \times S_1 + C \times S_1)) \times (S_2 \times \mathcal{P}(S_2 + C \times S_2 + C \times S_2)) \to \mathcal{P}(S_1 \times S_2)$$

sending $\langle\langle s_1, A_1\rangle, \langle s_2, A_2\rangle\rangle$ to

$$\{\langle s'_1, s_2\rangle | \kappa_{1,1} s'_1 \in A_1\} \cup \{\langle s_1, s'_2\rangle | \kappa_{2,1} s'_2 \in A_2\} \cup \bigcup_{c\in C} \left\{ \langle s'_1, s'_2\rangle \;\middle|\; \begin{array}{l} \kappa_{1,2}\langle c, s'_1\rangle \in A_1 \text{ and } \kappa_{2,3}\langle c, s'_2\rangle \in A_2 \\ \text{or} \\ \kappa_{2,2}\langle c, s'_2\rangle \in A_2 \text{ and } \kappa_{1,3}\langle c, s'_1\rangle \in A_1 \end{array} \right\},$$

where $\kappa_{i,j}$ is the j-th injection to the coproduct $S_i + C \times S_i + C \times S_i (i = 1, 2)$. The first and second set of the union describe the successors given by the transitions without input and output channels. The third set describes the successors given

by communications of two different automata, that synchronize two transitions with same input and output channels.

3.3 Probabilistic Synchronous Connector

Definition 3.7 A *probabilistic synchronous connector* psync is a connector on $\mathcal{D}$ with components

$$\mathsf{psync}_{\{S_i\}_{i\in I}} : \prod_{i\in I} \mathcal{D}(S_i) \longrightarrow \mathcal{D}(\prod_{i\in I} S_i) \tag{11}$$

sending $\langle \mu_i \rangle_{i\in I} \in \prod_{i\in I} \mathcal{D}(S_i)$ to their product probability distribution $\prod_{i\in I} \mu_i$.

3.4 Connectors with projections

We define a property of connectors.

Definition 3.8 Let γ be a connector from G to H. We say the connector γ *commutes with projections* if the following diagram with projections π_J and π'_J commutes for any finite family $\{S_i\}_{i\in I}$ of sets and its subfamily with $J \subseteq I$:

$$\begin{array}{ccc} \prod_{i\in I} G(S_i) & \xrightarrow{\gamma_{\{S_i\}_{i\in I}}} & H(\prod_{i\in I} S_i) \\ \Big\downarrow{\scriptstyle \pi'_J} & & \Big\downarrow{\scriptstyle H(\pi_J)} \\ \prod_{i\in J} G(S_i) & \xrightarrow[\gamma_{\{S_i\}_{i\in J}}]{} & H(\prod_{i\in J} S_i). \end{array} \tag{12}$$

And, under the assumption that every image of H is equipped with a partial order, we say the connector γ *weakly commutes with projections* if the above diagram weakly commutes. Here "weakly commutes" means that the image under the composition of the left down and the bottom is smaller than that of the top and the right down arrows.

We can observe the following:

Proposition 3.9 (i) *The synchronous connector* sync (8) *commutes with projections.*

(ii) *The* (C, V)*-asynchronous connector* $\mathsf{async}^{C,V}$ (9) *weakly commutes with projections, where the partial order on* $\mathcal{P}(X \times V)^V$ *for every set* X *is the pointwise order given by inclusion.*

(iii) *The probabilistic synchronous connector* psync (11) *commutes with projections.*

Proof. Routine. □

For the case of (C, V)-asynchronous connectors, the diagram (12) does not commute as follows:

Example 3.10 Consider the case of $(\emptyset, 1)$-asynchronous connector $\mathsf{async}^{\emptyset,1}$ (9) with index sets $I = \{1,2,3\}$ and $J = \{1,2\}$. For any $\langle\langle x, A\rangle, \langle y, B\rangle, \langle z, C\rangle\rangle \in \prod_{i\in\{1,2,3\}}(S_i \times \mathcal{P}(S_i))$,

$$\mathcal{P}(\pi_{\{1,2\}}) \circ \mathsf{async}^{\emptyset,1}_{\{S_1,S_2,S_3\}}(\langle\langle x, A\rangle, \langle y, B\rangle, \langle z, C\rangle\rangle) = \{\langle x, y\rangle\} \cup \{x\}\times B \cup A\times\{y\}.$$

On the other hand,

$$\mathsf{async}^{\emptyset,1}_{\{S_1,S_2\}} \circ \pi'_{\{1,2\}}(\langle\langle x, A\rangle, \langle y, B\rangle, \langle z, C\rangle\rangle) = \{x\}\times B \cup A\times\{y\}.$$

Hence the former image always has the pair $\langle x, y\rangle$ as an element, but the latter does not.

4 Coalgebraic Representations of Systems

We develop a coalgebraic representation of various systems by using the connectors defined in Section 3.

4.1 Synchronous Systems

Definition 4.1 A $\mathcal{P}$-coalgebra $M = (\prod_{i\in I} S_i, \alpha)$ over a finite family $\{S_i\}_{i\in I}$ of sets is called a *synchronous system* if there exists a family of state transition maps

$$\{f_i : \prod_{k\in I} S_k \longrightarrow \mathcal{P}_+(S_i)\}_{i\in I} \tag{13}$$

such that

$$\alpha = \left(\prod_{i\in I} S_i \xrightarrow{\langle f_i\rangle_{i\in I}} \prod_{i\in I} \mathcal{P}_+(S_i) \xrightarrow{\mathsf{sync}_{\{S_i\}_{i\in I}}} \mathcal{P}(\prod_{i\in I} S_i)\right).$$

Example 4.2 • Synchronous products of two nonterminating transition systems are examples of the synchronous systems.

• The synchronous systems of SMV [13] are examples of the synchronous systems.

4.2 Asynchronous Systems

Definition 4.3 A $\mathcal{P}_V$-coalgebra $M = (\prod_{i\in I} S_i, \alpha)$ over a finite family of sets $\{S_i\}_{i\in I}$ is called a (C, V)-*asynchronous system* if there exists a family of state transition maps

$$\{f_i : \prod_{k\in I} S_k \longrightarrow B_{C,V}(S_i)\}_{i\in I} \tag{14}$$

such that

$$\alpha = \left(\prod_{k\in I} S_k \xrightarrow{\langle\langle \pi_i, f_i\rangle\rangle_{i\in I}} \prod_{i\in I}(S_i \times B_{C,V}(S_i)) \xrightarrow{\mathsf{async}^{C,V}_{\{S_i\}_{i\in I}}} \mathcal{P}_V(\prod_{i\in I} S_i)\right).$$

Example 4.4 • The $(\emptyset, 1)$-asynchronous systems includes the asynchronous products (or interleaving products) of two transition systems, and the asynchronous systems of SMV [13].

- The (C, V)-asynchronous systems model some fragments of the modeling language in UPPAAL [15]. We can represent collection of automata that asynchronously update their states, and include the following features with them:
  · guarded transitions,
  · communications, that synchronise two transitions in different automata, through synchronisation labels on transitions,
  · shared data updated by the executions of transitions.

4.3 Probabilistic Synchronous Systems

We represent probabilistic synchronous systems as $\mathcal{D}$-coalgebras as follows:

Definition 4.5 A $\mathcal{D}$-coalgebra $M = (\prod_{i\in I} S_i, \alpha)$ over a finite family $\{S_i\}_{i\in I}$ of sets is called *a probabilistic synchronous system* if there exists a family of state transition maps

$$\{f_i : \prod_{k\in I} S_k \longrightarrow \mathcal{D}(S_i)\}_{i\in I} \tag{15}$$

such that

$$\alpha = \left(\prod_{k\in I} S_k \xrightarrow{\langle f_i\rangle_{i\in I}} \prod_{i\in I} \mathcal{D}(S_i) \xrightarrow{\mathsf{psync}_{\{S_i\}_{i\in I}}} \mathcal{D}(\prod_{i\in I} S_i) \right).$$

Example 4.6 The probabilistic synchronous system models Discrete Time Markov Chains used in PRISM [10].

5 A Representation of Reduction by the Cone of Influence

We represent the cone of influence reduction by using the coalgebraic representations of systems developed in Section 4.

5.1 Consistency of Transition maps

We define the notion of a consistent subset of an index set for the family of transition maps, which model variables that are independent from the outside. We describe a dependency analysis of state variables in Section 6 and see that a cone of influence is an instance of a consistent subset of the index set.

Definition 5.1 Let $\{f_i : \prod_{k\in I} S_k \to F(S_i)\}_{i\in I}$ be a family of state transition maps. A nonempty subset J of index set I is *consistent* with respect to the family if there is a family $\{g_i : \prod_{k\in J} S_k \to F(S_i)\}_{i\in J}$ of state transition maps on the subfamily

$\{S_i\}_{i\in J}$ such that the following diagram commutes with projections π_J and π'_J:

$$\begin{array}{ccc} \prod_{k\in I} S_k & \xrightarrow{\langle f_i\rangle_{i\in I}} & \prod_{i\in I} F(S_i) \\ \pi_J \downarrow & & \downarrow \pi'_J \\ \prod_{k\in J} S_k & \xrightarrow[\langle g_i\rangle_{i\in J}]{} & \prod_{i\in J} F(S_i). \end{array} \tag{16}$$

5.2 Cone of Influence Reduction

For these systems with coalgebraic representations, the reduction by a cone of influence can be explained schematically as follows:

Main Theorem 1 *Let $M = (\prod_{i\in I} S_i, \alpha)$ be an H-coalgebra specified by a family $\{f_i : \prod_{k\in I} S_k \to G(S_i)\}_{i\in I}$ of state transition maps for functor G and a connector γ from G to H, i.e., the structure map α is given by the composite:*

$$\prod_{k\in I} S_k \xrightarrow{\langle f_i\rangle_{i\in I}} \prod_{i\in I} G(S_i) \xrightarrow{\gamma_{\{S_i\}_{i\in I}}} H(\prod_{i\in I} S_i).$$

If the the connector γ commutes with projections, then, for any consistent subset $J \subseteq I$ with respect to the collection of state transition maps, there exists a reduced H-coalgebra $M_J = (\prod_{i\in J} S_i, \alpha_J)$ such that the projection $\pi_J : \prod_{i\in I} S_i \to \prod_{i\in J} S_i$ gives a morphism of H-coalgebras from M to M_J.

If we assume every image of H is equipped with a partial order and the connector γ weakly commutes with projections, then the projection π_J gives an oplax morphism from M to M_J.

Proof. Since the subset $J \subseteq I$ is consistent with respect to the family of state transition maps, we have a collection $\{g_i : \prod_{k\in J} S_k \to G(S_i)\}_{i\in J}$ of state transition maps over the subfamily $\{S_i\}_{i\in J}$. Define

$$\alpha_J = \left(\prod_{k\in J} S_k \xrightarrow{\langle g_i\rangle_{i\in J}} \prod_{i\in J} G(S_i) \xrightarrow{\gamma_{\{S_i\}_{i\in J}}} H(\prod_{i\in J} S_i) \right).$$

Combining the diagram (16) in Definition 5.1 and the commutative diagram (12) in Definition 3.8, we obtain the following commutative diagram:

$$\begin{array}{ccccc} \prod_{k\in I} S_k & \xrightarrow{\langle f_i\rangle_{i\in I}} & \prod_{i\in I} G(S_i) & \xrightarrow{\gamma_{\{S_i\}_{i\in I}}} & H(\prod_{i\in I} S_i) \\ \pi_J \downarrow & & \pi'_J \downarrow & & \downarrow H(\pi_J) \\ \prod_{k\in J} S_k & \xrightarrow[\langle g_i\rangle_{i\in J}]{} & \prod_{i\in J} G(S_i) & \xrightarrow[\gamma_{\{S_i\}_{i\in J}}]{} & H(\prod_{i\in J} S_i). \end{array} \tag{17}$$

Hence the projection $\pi_J : \prod_{k\in I} S_k \to \prod_{k\in J} S_k$ gives a morphism of H-coalgebras from M to M_J.

If we assume the connector γ weakly commutes with projections, then the above diagram (17) weakly commutes, and the projection π_J gives an oplax morphism from M to M_J. □

Hence we obtain, by Proposition 2.6, 3.9 and Lemma 2.7, the following COI results for systems with coalgebraic representations in Section 4:

Theorem 5.2 *Under the same assumption as Main Theorem 1:*

(i) *For the case of synchronous systems, we have a morphism of $\mathcal{P}$-coalgebra from M to M_J, hence, a bisimulation $\mathsf{Graph}(\pi_J)$ between M and M_J.*

(ii) *For the case of probabilistic synchronous systems, we have a morphism of $\mathcal{D}$-coalgebra from M to M_J, hence, a probabilistic bisimulation $\mathsf{Graph}(\pi_J)$ between M and M_J.*

(iii) *For the case (C, V)-asynchronous systems, we have an oplax morphism of $\mathcal{P}_V$-coalgebras from M to M_J, hence, a simulation $\mathsf{Graph}(\pi_J \times \mathrm{id}_V)^{-1}$ between $\mathcal{P}$-coalgebras that are exponential adjuncts of M_J and M.*

Remark 5.3 For practical application of COI in verification, we actually need bisimulations or simulations between Kripke structures. We can easily extend the above results for Kripke structures by considering suitable labeling functions.

6 Dependency Graph

In the previous sections, given a family of state transition maps, we have seen that the cone of influence reduction is possible if we have a consistent subset with respect to the family. Here, we show that the consistent subsets can be obtained by analysing a "dependency graph" of the family of state transition maps.

Definition 6.1 Let $\{f_i : \prod_{k\in I} S_k \to F(S_i)\}_{i\in I}$ be a family of state transition maps. A state transition map $f_i : \prod_{k\in I} S_k \to F(S_i)$ *depends on a subset* $J \subseteq I$ when there is a map $g_i : \prod_{k\in J} S_k \to F(S_i)$ such that f_i factors through projection and g_i, i.e., $f_i = g_i \circ \pi_J$ holds.

If a state transition map $f_i : \prod_{k\in I} S_k \to F(S_i)$ depends on a subset J of I, then the f_i also depends on any subset that includes J. Hence, for each state transition map f_i, there is a minimal subset on which f_i depends.

Example 6.2 Consider the following fragment of SMV code:

```
init(x) := 0;
init(y) := 0;
init(z) := 0;
next(x) := Case
    x = 0 & y = 0: 1;
    x = 1 & z = 0: {0,1};
```

```
    1 : 0;
next(y) := Case
    y = 0: 1;
    y = 1: {0,1};
    1 : 0;
next(z) := Case
    z = 0 & y = 0: 1;
    y = 1: 0;
    1 : 0;
```

The state transition maps of variables x, y, and z depend on the sets $\{\mathtt{x},\mathtt{y},\mathtt{z}\}$, $\{\mathtt{y}\}$, and $\{\mathtt{y},\mathtt{z}\}$, respectively.

Definition 6.3 Let $\{f_i : \prod_{k\in I} S_k \to F(S_i)\}_{i\in I}$ be a family of state transition maps. For each map f_i in the collection, we fix a subset $C_i \subseteq I$ to be a minimal subset on which f_i depends. A *dependency graph for the family* $\{f_i : \prod_{k\in I} S_k \to F(S_i)\}_{i\in I}$ *of state transition maps* is a directed graph (I, E) consisting of the index set I and a set of edges $E \subseteq I \times I$ given by

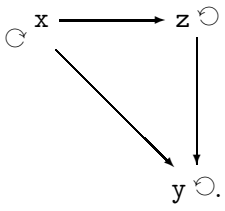
$$(k, j) \in E \quad \text{iff} \quad j \in C_k.$$

Proposition 6.4 *Let (I, E) be a dependency graph for a family of state transition maps $\{f_i : \prod_{k\in I} S_k \to F(S_i)\}_{i\in I}$. Every nonempty subset $J \subseteq I$ which is closed under the relation E, namely $\{j \mid \exists i \in J.(i, j) \in E\} \subseteq J$ holds, is consistent with respect to the collection.*

Example 6.5 Let (I, E) be a dependency graph. For any $J \subseteq I$, define a subset $\mathrm{reach}(J) \subseteq I$ to be a set of all reachable nodes from J. Then $\mathrm{reach}(J)$ is closed under the relation E, and hence consistent. This set $\mathrm{reach}(J)$ corresponds to the cone of influence for variables in J [3].

Example 6.6 For the case of the synchronous product in Example 4.2, and the asynchronous product in Example 4.4, the dependency graph has just two nodes with self-returning edges for each node. Hence, any non-empty subset of the index set is consistent.

Example 6.7 For the case of SMV code in Example 6.2, the dependency graph is given by



We have three consistent subsets: $\{\mathtt{y}\}$, $\{\mathtt{y},\mathtt{z}\}$, and $\{\mathtt{x},\mathtt{y},\mathtt{z}\}$.

7 Nondeterministic Choice Constructions that Preserve Cone of Influence Reduction

It is not practical to represent the system only in the previous framework for synchronous, (C, V)-asynchronous, or probabilistic synchronous systems. For that reason, we introduce additional constructions that preserve cone of influence reduction. These constructions recall a nondeterministic choice for the invocation of the modules in SMV.

We remark that these constructions preserve the cone of influence reductions if every component has the same consistent subset.

7.1 Nondeterministic Choice

This construction composes two $\mathcal{P}$-coalgebras with same carrier, and at every instance the composed system updates its state obeying either structure map.

We define the nondeterministic choice composition as follows:

Definition 7.1 Let $M_1 = (X, \alpha_1)$ and $M_2 = (X, \alpha_2)$ be two $\mathcal{P}$-coalgebras with the same carrier X. A *nondeterministic choice composition* of M_1 and M_2 is a $\mathcal{P}$-coalgebra $M_1 \vee M_2 = (X, \alpha)$ with the same carrier X and the structure map $\alpha : X \to \mathcal{P}(X)$ given by the composite

$$X \xrightarrow{\langle \alpha_1, \alpha_2 \rangle} \mathcal{P}(X) \times \mathcal{P}(X) \xrightarrow{\mathsf{or}_X} \mathcal{P}(X),$$

where the map $\mathsf{or}_X : \mathcal{P}(X) \times \mathcal{P}(X) \to \mathcal{P}(X)$ is defined by

$$\langle A, B \rangle \mapsto A \cup B.$$

The composition has the following property:

Proposition 7.2 *Let M_1, M_2 be $\mathcal{P}$-coalgebras with the same carrier X, and N_1, N_2 $\mathcal{P}$-coalgebras with the same carrier Y. If a map $f : X \to Y$ gives morphisms/lax morphisms/oplax morphisms of $\mathcal{P}$-coalgebras $f : M_1 \to N_1$ and $f : M_2 \to N_2$, then again f gives a morphism/lax morphism/oplax morphism $f : M_1 \vee M_2 \to N_1 \vee N_2$.*

This property expresses that the composition preserves the COI reduction in the following sense:

Example 7.3 Consider, for example, a reduction of a nondeterministic choice composition $M_1 \vee M_2$ of two synchronous systems $M_1 = (\prod_{i \in I} S_i, \alpha_1)$ and $M_2 = (\prod_{i \in I} S_i, \alpha_2)$ with the same carrier. If we assume a subset $J \subset I$ is consistent with respect to both families of state transition maps specifying M_1 and M_2, then, by Theorem 5.2, there are reduced synchronous systems $M_1' = (\prod_{i \in J} S_i, \alpha_1')$ and $M_2' = (\prod_{i \in J} S_i, \alpha_2')$ such that the projection $\pi_J : \prod_{i \in I} S_i \to \prod_{i \in J} S_i$ gives two morphisms of $\mathcal{P}$-coalgebras from M_1 to M_1' and from M_2 to M_2'. Under these assumptions, by Proposition 7.2, the nondeterministic choice composition $M_1' \vee M_2'$ yields a reduced $\mathcal{P}$-coalgebra of the $M_1 \vee M_2$ since the projection π_J gives again a morphism of $\mathcal{P}$-coalgebra from $M_1 \vee M_2$ to $M_1' \vee M_2'$.

7.2 Nondeterministic Choice for Probabilistic Systems

We give probabilistic version of a nondeterministic choice composition in this section. This construction composes $\mathcal{D}$-coalgebras with same carrier.

Definition 7.4 Let $M_1 = (X, \alpha_1), \ldots, M_n = (X, \alpha_n)$ be $\mathcal{D}$-coalgebras with the same carrier X. A *probabilistic nondeterministic choice composition* of n $\mathcal{D}$-coalgebras $M_1, \ldots, M_n$ is a $\mathcal{D}$-coalgebra $M_1 \vee \cdots \vee M_n = (X, \alpha)$ with the same carrier X and a structure map $\alpha : X \to \mathcal{D}(X)$ given by the composite

$$X \xrightarrow{\langle \alpha_1, \ldots, \alpha_n \rangle} \mathcal{D}(X) \times \cdots \times \mathcal{D}(X) \xrightarrow{\mathsf{por}^n_X} \mathcal{D}(X),$$

where $\mathsf{por}^n_X : \mathcal{D}(X) \times \cdots \times \mathcal{D}(X) \to \mathcal{D}(X)$ is defined by

$$\langle \mu_1, \ldots, \mu_n \rangle \mapsto \sum_{k=1}^{n} \frac{1}{n} \mu_k.$$

We have the following preservation result for this composition:

Proposition 7.5 *Let $M_1, \ldots, M_n$ be $\mathcal{D}$-coalgebras with the same carrier X, and $Q_1, \ldots, Q_n$ $\mathcal{D}$-coalgebras with the same carrier Y. If a map $f : X \to Y$ gives a morphism of $\mathcal{D}$-coalgebras $f : M_i \to Q_i$ for every $1 \leq i \leq n$, then again f gives a morphism $f : M_1 \vee \cdots \vee M_n \to Q_1 \vee \cdots \vee Q_n$ of $\mathcal{D}$-coalgebras.*

8 Conclusion

In this paper we have presented a coalgebraic representation of reduction by the cone of influence. We have developed coalgebraic representations of various systems given by compositions of state transition maps and connectors. These representations include synchronous systems, asynchronous systems, asynchronous systems with message passing by channels, and those with shared variables, and probabilistic synchronous systems. We have demonstrated COI results can be schematically obtained by the analysis of dependency graphs and a property of connectors, which give a framework for providing the technique. We hope this coalgebraic framework is helpful and has application in practical verification.

Acknowledgment

The authors would like to thank the anonymous referees for their comments and suggestions.

References

[1] Armin Biere , Edmund Clarke , Richard Raimi , Yunshan Zhu, *Verifying Safety Properties of a PowerPC$^{\text{TM}}$ Microprocessor Using Symbolic Model Checking without BDDs*, Lecture Notes in Computer Science, vol 1633, pp. 60–71, 1999.

[2] E.M. Clarke, O. Grumberg, D.E. Long, *Model Checking and Abstraction*, ACM Trans. Program. Lang. Syst., Vol.16, 1512–1542, 1994.

[3] E.M. Clarke, O. Grumberg and D. Peled, *Model Checking*, The MIT Press, 1999.

[4] H. Hansson, B. Jonsson. *A Logic for Reasoning About Time and Probability*, Formal Aspects of Computing 6(5), 512-535, 1994.

[5] Gerald J. Holzmann, *The SPIN Model Checker*, Addison Wesley Professional, 2004.

[6] Robert P. Kurshan, *Computer-Aided Verification of Coordinating Processes: The Automata-Theoretic Approach*, Princeton University Press, 1995.

[7] Marta Kwiatokowska, Gethin Norman, and David Parker, *Modelling and Verification of Probabilistic Systems*, in Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems, P. Panangaden and F. van Breugel (eds.), American Mathematical Society, 2004, vol. 23, CRM Monograph Series.

[8] Kim Guldstrand Larsen, Arne Skou *Bisimulation through Probabilistic Testing*, Inf. Comput. 94(1): 1-28 (1991).

[9] NuSMV: a new symbolic model checker, URL: http://nusmv.irst.itc.it/

[10] Probabilistic Symbolic Model Checker, URL: http://www.cs.bham.ac.uk/~dxp/prism/

[11] J. Rutten, *Universal coalgebra: a theory of systems*, Theoretical Computer Science, Vol. 249, 3–80, 2000.

[12] Roberto Segala and Nancy Lynch, *Probabilistic Simulations for Probabilistic Processes*, Nordic Journal of Computing, 2(2):250-273, 1995.

[13] The SMV System, URL: http://www-2.cs.cmu.edu/~modelcheck/smv.html

[14] ON-THE-FLY, LTL MODEL CHECKING with SPIN, URL: http://spinroot.com

[15] UPPAAL Home, URL: http://www.uppaal.com

[16] E.P. de Vink, J.J.M.M. Rutten, *Bisimulation for probabilistic transition systems: a coalgebraic approach*, Theoretical Computer Science, Vol. 221, 271–293, 1999.

[17] Dong Wang, Pei-Hsin Ho, Jiang Long, James Kukula, Yunshan Zhu, Tony Ma, Robert Damiano, *Formal Property Verification by Abstraction Refinement with Formal, Simulation and Hybrid Engines*, 38th Design Automation Conference (DAC'01), 6, June, 2001.